OIPE JC57 JUL 08 2004 PATENT & TRADEMARK OFFICE

EXPRESS MAIL LABEL NO. EV375067548US
Attorney Docket No. 3175-51A
(1660/101)

DUAL MODE DATA COMPRESSION TECHNIQUE

RELATED APPLICATIONS

[0001] This application is related to U.S. application Ser. No. ~~———,~~09/750,188, (Attorney Docket: 3175-51), entitled ~~"~~"Enhanced Data Compression Technique~~"~~" and filed concurrently herewith on Dec. 29, 2000.

TECHNICAL FIELD

[0002] The present application relates generally to data compression and more particularly to an enhanced data compression technique. This technique is particularly suitable for use in the graphic~~al~~ arts for compressing large images.

BACKGROUND ~~ART~~OF THE INVENTION

[0003] In the graphic arts there is a tendency to have extremely large, one-bit-per-sample images approaching or even exceeding 2 gigabytes of data. The need to compress such data has been well known for many years.

[0004] One proposed technique for compressing such data is commonly referred to as a ~~pack-bit~~ ~~(~~PackBits, hereinafter "PB~~)~~," compression technique. ~~Using proposed~~ The PB compression ~~techniques,~~technique produces either a string of characters ~~is~~ preceded with a count and a repeat character code or, alternatively, a single byte pattern ~~is~~ preceded with a count. ~~Proposed~~The PB compression techniques ~~are~~is capable of processing data very quickly. ~~These techniques~~ This technique also provides satisfactory results ~~if~~when the data is ~~either~~a string of solid black or solid respectively. Accordingly, the PB technique~~s~~ provides reasonably satisfactory results for non-color image data.

[0005] An exemplary ~~pack-bits~~PackBits representation of a stream of sequential input data, as it would appear entering a processor prior to encoding, might include the string of characters ~~"~~“abc0000000000~~"~~”. Using the PB technique, the processor would first ~~determined whether or not the~~determine if a first character ~~"~~“a~~"~~” and ~~the~~a second character ~~"~~“b~~" match~~” are the same character. ~~Under~~In some proposed PB techniques, the processor might scan ahead to consider ~~other matches in certain of the~~ subsequent characters when determining if a stream contains the same repeated character. In ~~any event, since in~~ the present example, the ~~determination is~~comparison that determines if “a” and “b” are the same character returns a negative~~,~~ ~~the~~result. The processor then proceeds to encode the input data as a literal string with a length. Next ~~T~~the processor ~~next~~ determines ~~whether or not~~if the second character ~~"~~“b~~"~~” and the third character ~~"~~“c~~" match~~” are the same character. Since this determination is also negative, the processor will proceed to encode the three characters of the input data as a literal string with a length. The processor ~~now~~then determines ~~whether or not~~if the third character ~~"~~“c~~"~~” and the fourth character ~~"~~“0~~" match~~” are the same character. Since this determination is also negative, the processor will proceed to encode the four characters of the input data as a literal string with a length. The processor continues ~~by determining whether or not~~and determines if the fourth character ~~"~~“0~~"~~” and the fifth character ~~"~~“0~~" match~~”are the same character. Since this determination is positive, the processor continues by repeatedly determining ~~whether or not~~if the immediately subsequent characters in the sequence are also ~~match,~~the same character until it makes a negative determination. The processor thereby determines the repeat count for the character ~~"~~“0~~"~~”. Based on the initial positive determination, the processor also proceeds to encode the first three characters of the input data sequence, i.e. ~~"~~“a~~"~~,~~"~~””b~~"~~” and ~~"~~“c~~"~~,” as a literal string with a length and the following 10 characters of the input data sequence, i.e. the ~~"~~“0~~"~~” . . . ~~"0",~~“0,” as a repeat character with a count.~~-~~

[0006] Accordingly, the processor generates encoded output data forming a 2-byte sequence including the strings of characters "“82abc"” and "“090"”. In the output data, the "“8"” serves as a header ~~and indicates~~indicating that the total length of the sequence is 8 bits and that a literal string follows~~,~~. ~~t~~The ~~"~~ “2"” indicates that the length of the literal string is three characters, i.e. characters "“a",” "“b",” and "“c"~~,~~”. ~~t~~The first "“0"” indicates that a repeat character follows, and the "“9"” indicates that the repeat character ~~represented by~~, the second ~~"0"~~“0,” is repeated 10 times. Using one-off numbers such as the “2” to indicate a literal string of 3 characters, and the “9” to indicate that a repeat character is repeated 10 times. Using one-off numbers such as the “2” to indicate a literal string of 3 characters, and the “9” to indicate that a repeat character is repeated 10 times, is efficient because 129 bytes can be packed using number up to 128.

[0007] To decode the encoded sequence "“82abc~~090",~~090,” the receiving processor first reads the ~~new~~ header ~~"8",~~“8,” which is the highest order bit~~, and from~~. From the header, the processor determines that a literal string follows. The processor then extracts the string length, ~~"2"~~“2,” and reads the next three characters "“a",” "“b"” and "“c"”. At this stage, the output string is “abc” and the remaining input string is “090.” The processor ~~next~~ then reads the first "“0~~" and from this determines~~” which indicates that a repeat character follows. The processor continues by extracting the repeat count, ~~"9"~~“9,” and ~~reading~~then reads the next character ~~"0", which is~~“0,” the character to be repeated 10 times. ~~It will be recognized that by using one-off numbers such as the "2" to indicate a literal~~ The resulting decoded string ~~of 3 characters and~~is “abc0000000000,” is the ~~"9"~~string originally presented prior to ~~indicate that a repeat character is repeated 10 times, a close to 1% improvement is obtainable because 128 bytes can be packed into 129.~~encoding.

[0008] As should be clear from the above, the PB technique~~s~~ processes only one character at a time. Accordingly, PB ~~techniques are~~is incapable of compressing strings of repeating multi~~ple~~-byte patterns of characters. The PB technique~~s~~ also ha~~ve~~s a relatively limited compression rate, generally no more than 64 to 1. Thus, the PB compression technique~~s~~ provides unsatisfactory results when used to compress color image data, which typically contains repeating multi-byte patterns of characters instead of repeating single-byte 0s and 1s.

[0009] Another proposed technique for compressing image data is ~~commonly referred to as~~ the Lempel-Ziv-Welch ~~(~~, hereinafter "LZW~~)~~," compression technique. Using ~~proposed~~the LZW compression ~~techniques~~technique, variable length ~~of~~ strings of byte based data can be processed. ~~Proposed~~The LZW compression technique~~,~~ processes the data somewhat slower than the PB compression technique~~s~~, but provides satisfactory results on data representing both color images ~~as well as~~and black -and -white images. However, since these techniques are based on single bytes of data, such techniques are incapable of compressing data on an arbitrary pixel or bit boundary basis. Additionally, ~~although such techniques, are~~though LZW is capable of providing a higher compression rate than PB, LZW's compression ~~techniques, LZW techniques~~rate is still ~~offer a~~ somewhat limited ~~compression rate~~.

~~An exemplary LZW representation of a stream of sequential input data, as it would appear entering a processor prior to encoding, might include the string of characters "abc01c01".~~ **[0010]** Using the LZW technique, the encoding and decoding processors must coordinate ~~on~~ the transmission and receipt of codes. The LZW technique~~s~~ uses a compression dictionary containing some limited number of compression codes defined during the processing of the input data. The characters in the input string are read on a character by character basis to determine if a sub-string of characters match a compression code defined during the processing of prior characters in the input string. If ~~so~~a pattern match is found, the matching sub-string of characters ~~are~~is encoded with the applicable compression code. If a sub-string of characters does not match a pre-existing code, a new code corresponding to the sub-string is added to the dictionary. Sub-strings are initially defined by codes having 9 bits ~~or digits~~, but the number of bits may be increased up to 12 bits to add new codes. Once the 12 -bit limit is exceeded, the dictionary is reset and subsequent codes are again defined initially with 9 bits. In conventional implementations of the LZW technique~~s~~, two codes are predefined, i.e. defined prior to initiating processing of the input string. In the present example these codes are the code 100, representing a "reset," and the code 101, representing an "end." In the present example, the codes 102, 103, ~~and 104~~104, etc. represent strings of new patterns that are identified during the processing of the input data.

[0011] An exemplary LZW representation of a stream of sequential input data, as it would appear entering a processor prior to encoding, might include the string of characters “abc01c01”. Using the LZW technique, the encoding processor ~~would~~ first reads the “a” in the sequence and the “b” immediately thereafter~~ in the sequence~~. The processor then determines if a code exists for the character sequence “ab”. Since, in this example, no such code exists at this point in the processing, a new code 103 is generated to represent the new pattern string “ab” and is added to the existing code dictionary. The processor continues by reading the “c” immediately following the “b” in the sequence. The processor determines if a code exists for the character sequence “bc”. Since, in this example, no such code exists at this point in the processing, a new code 104 is generated to represent the new pattern string “bc” and the code is added to the code dictionary.

[0012] The processor continues ~~by~~and read~~ings~~s the “0” immediately following the “c” in the sequence. The processor determines if a code exists for the character sequence “c0”. Since, in this example, no such code exists at this point in the processing, a new code 105 is generated to represent the new pattern string “c0” and the code is added to the code dictionary. The processor continues ~~further by reading~~and reads the “1” immediately following the “0” in the sequence. The processor determines if a code exists for the character sequence “01”. Since, in this example, no such code exist at this point in the processing, a new code 106 is generated to represent the new pattern string “01”and the code is added to the dictionary. The processor proceeds ~~by~~and read~~ings~~s the “c” immediately following the “1” in the sequence. The processor determines if a code exists for the character sequence “1c”. Since, in this example, no such code exists at this point in the processing, a new code 107 is generated to represent the new pattern string “1c” and the new code is added to the dictionary.

[0013] The processor proceeds by reading the “0” immediately following the second “c” in the sequence. The processor determines if a code exists for the character sequence “c0”. In this example, such a code, i.e. code 105, does exist. The processor therefore~~ proceeds by reading,~~ determines if a longer pattern match can be made, and reads the “1” immediately following the second “c0” in the sequence. The processor determines if a code exists for the character sequence ~~"~~“c01~~"~~”. Since, in this example, such a code does not exist, a new code 108 is generated to represent the new pattern string ~~"~~“c~~01"~~01,” which can also be represented as ~~"~~“1051~~"~~”. The processor ultimately generates encoded output data ~~forming a~~ that forms the sequence ~~including the string~~ of characters: ~~"~~“100abc01c1051”. The sequence, broken down into symbols represents: a reset (100); a literal string (abc01c); a previously found pattern (105~~"~~); and a literal (1).

[0014] Using the LZW technique, the encoding processor builds a tree of codes generated using other codes. This is a primary reason why the LZW technique~~s~~ provides satisfactory results even though processing is performed on a byte by byte basis to find repeating ~~bytes.~~ byte patterns. That is, the downstream encoding builds on the upstream encoding. However, using the LZW technique, the encoding processor can take significant processing time to encode large sequences. For example, if there is a large~~, say a megabyte,~~ occurrence of adjacent 0's or 1's, a significant period of time will be required by the processor to encode the sequence.

[0015] The decoding processor builds a similar tree from the codes received from the encoding processor. Basically, the decoding processor performs the reciprocal of the encoding process to decode the encoded sequence characters ~~"~~“100abc011051~~"~~”.

[0016] In summary, the PB compression technique is deficient in that it addresses only single byte repeats and is limited to a 64 to 1 compression rate. Therefore, it is not suitable for color images. On the other hand, while the LZW compression technique addresses multi-byte repeats and has a compression rate of perhaps 500 to 1, ~~but~~it requires significant processing time to build the codes ~~which~~that are required to obtain good compression. Hence, although the LZW technique may be suitable ~~where~~for encoding relatively small amounts of data ~~are involved~~, ~~where the~~when encoding ~~of~~ gigabytes of data ~~is required~~, such as ~~with~~ an 80 inch ~~times.~~ x 50 inch image having 2400 dots per inch, the processing time and/or resources necessary to encode data ~~using the LZW technique~~ make using the LZW technique alone impractical.

[0017] Accordingly, ~~a~~the need exists for a technique which can quickly compress large amounts of image data, offer a still higher compression rate than previously proposed techniques, and provide satisfactory results when used to compress either color or non-color image data.

~~OBJECTIVES~~OBJECTIVE SUMMARY OF THE INVENTION

[0018] It is an object of the present the invention ~~to provide~~provides a technique for quickly compressing large amounts of image data.

[0019] It is a further object of the present the invention ~~to provide~~provides a technique which facilitates high compression rates for ~~either~~both color ~~or~~and non-color image data.

~~It is yet another object of the present~~[0020] The invention ~~to provide~~provides a technique ~~which~~that gives satisfactory results when used to compress ~~either~~both color ~~or~~and non-color image data~~. Additional objects, advantages, novel features of the present invention will become apparent to those skilled in the art from this disclosure, including the following detailed description, as well as by practice of the invention. While the invention is described below with reference to preferred embodiment(s), it should be understood that the invention is not limited thereto. Those of ordinary skill in the art having access to the teachings herein will recognize additional implementations, modifications, and embodiments, as well as other fields of use, which are within the scope of the invention as disclosed and claimed herein and with respect to which the invention could be of significant utility~~.

SUMMARY DISCLOSURE OF THE INVENTION

[0021] According to one embodiment of the~~ present~~ invention, an encoder for compressing image information ~~includes~~comprises a memory and a processor. The memory is configured to store a sequence of characters representing an image. The processor is configured to determine if the stored sequence of characters corresponds to either a banded image, such as a segment or slice across the entire image, or a page image, such as one of multiple separate images making up the entire image.~~The processor operates in a first mode to encode the stored sequence of characters, if~~ If the stored sequence of characters is determined to correspond to ~~the~~a banded image~~.~~, ~~T~~the processor operates in a ~~second mode, different than the~~ first mode~~,~~ to encode the stored sequence~~ of characters, if~~. If the stored sequence of characters is determined to correspond to ~~the~~a page image, the processor operates in a second mode to encode the stored sequence of characters.

[0022] Preferably, when the processor ~~encodes the stored sequence of characters in accordance with a pack-bit compression technique~~is operating in the first mode~~ of operation~~, PackBits compression is used, and when the processor is operating in ~~accordance with an~~the second mode, LZW compression ~~technique~~is used by default. However, if while in the second mode~~ of operation. Beneficially~~, the processor ~~is also configured to encode the stored sequence of characters in accordance with a pack-bit~~determines that PackBits compression ~~technique in~~is appropriate, e.g. when presented with a string of repeating 0s or 1s, the ~~second mode of operation,~~processor may switch to using PackBits as the compression technique. Making this ~~can be beneficial in some cases~~transition between compression techniques does not change the mode from second to first. The mode remains the same, only which compression technique is used is altered.

[0023] Advantageously, ~~if the stored sequence~~during operation of ~~characters is determined to correspond to~~ the ~~page~~second ~~image~~mode, the processor ~~is~~can be further configured to determine if the stored sequence of characters corresponds to a primarily white page image or a primarily black page image. For example, ~~which~~that might be the case for a template ~~type~~page, if the page is primarily white or primarily black. The processor encodes the stored sequence of characters using a PackBits compression technique. If the page image~~. If so~~ is neither primarily black nor primarily white, the processor encodes the stored sequence of characters ~~in accordance~~ with ~~a first compression technique, e.g. a pack-bit compression technique, while operating in~~ the ~~second mode of operation. If not, the processor encodes the stored first sequence of characters in~~

~~accordance with a second compression technique, different than the first compression technique, e.g. an~~ LZW compression technique~~, while operating in the second mode of operation~~.

[0024] In one practical implementation, an imaging system may include a raster image processor which determines if a sequence of characters corresponds to a banded image or a page image. ~~The~~ If the sequence of characters is determined to correspond to a banded image, the raster image processor then operates in ~~a~~the first mode to encode the sequence of characters ~~if~~. If the sequence of characters is determined to correspond to ~~the~~a ~~banded~~page image, ~~and to~~ the processor operates in a second mode, different than the first mode, to encode the sequence of characters ~~if the sequence of characters is determined to correspond to the page image~~.

[0025] An imager controller receives the encoded sequence of characters. The imager controller then operates in either ~~a~~the first mode or the second mode to decode the received encoded sequence of characters back into the unencoded sequence of characters. More particularly, ~~the controller operates in a first mode~~ if the encoded sequence of characters corresponds to ~~the~~a banded image, ~~and~~the controller operates in ~~a~~the ~~second~~first mode ~~if~~. If the encoded sequence of characters corresponds to ~~the~~a page image, the controller operates in the second mode.

[0026] Preferably, in the first mode of operation, the raster image processor encodes the sequence of characters ~~in accordance with~~using a ~~pack bit~~PackBits compression technique ~~in~~. In the ~~first~~second mode of operation, ~~and in accordance with an~~the raster image processor uses the LZW compression technique ~~in the second mode of operation~~by default. Beneficially, the raster image processor is also capable of encoding the sequence of characters ~~in accordance with~~using a ~~pack bit~~PackBits compression technique in the second mode of operation if appropriate.

[0027] In ~~accordance with~~ other ~~aspects of the invention~~embodiments, while operating in the second mode, if the ~~first~~ sequence of characters is determined to correspond to ~~the~~a page image, the raster image processor then determines if the sequence of characters corresponds to ~~a~~an image that is primarily white ~~page image~~ or ~~a~~ primarily black ~~page image~~. If so, the raster image processor encodes the sequence of characters in ~~accordance with~~using a first compression technique, for example, such as a ~~pack-bit~~PackBits technique~~, while operating in the second mode of operation~~. If ~~not~~the image is neither primarily black nor primarily white, the raster image processor encodes the sequence of characters ~~in accordance with~~using a second compression technique~~ which is different than~~, for example, the~~ first compression technique, such as a~~ LZW technique~~, while operating in the second mode of operation~~.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] FIG. 1 depicts an exemplary simplifie~~s~~d depiction of an image processing system in accordance with a first embodiment of the present invention.

[0029] FIG. 2 depicts an exemplary simplifie~~s~~d depiction of an image processing system in accordance with a second embodiment of the present invention.

[0030] FIG. 3 depicts an exemplary code dictionary in accordance with the second embodiment of the present invention.

[0031] FIG. 4 and FIG. 5 are flowcharts of exemplary embodiments of the invention.

~~BEST MODE FOR CARRYING OUT~~DETAILED DESCRIPTION OF THE INVENTION

[0032] In pre-press imaging, particularly for~~ the~~ flats having an entire plate worth of image information, most of the data is often either solid black or solid white, ~~and hence~~ digitally represented in binary form by ~~all~~a stream of repeating 1's or 0's~~.~~, respectively. For halftone images all of the data is black and white, i. e., 1's and 0's.

~~FIG.~~ [0033 FIG. 1 is a~~ somewhat~~ simplified, exemplary depiction of a image processing system 1000 according to ~~a~~the first embodiment of the present invention. The system 1000 includes a raster image processor~~ (~~, hereinafter "RIP~~)~~," 1050, ~~which~~an imager controller 1100, and an imager 1150. The raster processor includes a processor 1050a and a memory 1050b for storing processing instructions and other data~~ as required~~. The RIP 1050 receives image data and converts the image data into encoded data. The ~~image~~encoded data is then transmitted to ~~an~~the imager ~~control processor~~controller 1100, which includes a processor ~~1000~~1100a and a memory 1100b for storing processing instructions and other data~~ as required~~. The imager controller 1100 generates imager control signals ~~to the imager 1150 in accordance with~~based on the data received from the RIP ~~1050, to~~1050. The imager controller 1100 sends the imager control signals to the imager 1150.~~ More particularly~~ Specifically, the control signals from the processor 1100a control the operation of the imager scanning assembly 1150a ~~so as~~ to form the image on a medium 1150c, such as, a film or plate, supported within the imager 1150. As shown, the imager includes 1150 uses a cylindrical drum 1150b ~~for~~to support~~ing~~ the medium 1150c. Alternatively, ~~but~~the imager 1150 could ~~alternatively include~~use a flat bed or external drum for supporting the medium.

[0034] In a first mode of operation, which will hereafter be referred to as a flat banding "banded mode," the RIP 1050 receives an 80 inch~~.times.~~ x 50 inch color separated image having 2400 dots per inch.~~ Preferably~~ The image could, for example, correspond to multiple pages of a magazine. In such a case, using imposition software on a front end preprocessor (not shown), the image could~~, for example, correspond to multiple pages of a magazine. In such a case, the image is~~ be formatted such that the image printed from the imaged medium 1150c is positioned ~~so as~~ to facilitate cutting, folding, and stitching to create multiple properly printed and positioned magazine pages.~~ In any event, the~~ The RIP 1050 converts the entire image into multiple gigabytes of data encoded data as a single job.

[0035] However, due to processing power limitations of RIP 1050, the entire image cannot be converted into encoded data in a single operational process. ~~Accordingly~~Instead, the RIP 1050 slices the image ~~is sliced~~ into bands~~,~~ prior to the data being converted~~, typically by the~~. The RIP ~~1050. If converted by the RIP 1050, this~~ processor 1050a may perform the banding of the image~~ may be performed by the RIP processor 1050~~ or a~~. However, conversion could also be preformed outside the RIP, for example by a~~ preprocessor (not shown) could also perform the banding of the image. In the preferred embodiment, the RIP processor 1050a ~~converts~~encodes the image data ~~representing~~in each of the bands ~~into encoded data~~ in a separate operational process. Thus, the job of encoding all the image bands, and therefore the entire image, is completed only after the RIP 1050 performs multiple, separate operational processes ~~are performed by the RIP 1050 so as to convert all of the image data representing the bands for the entire image into encoded data~~. In practice, the larger the image, the smaller ~~is~~ each image band is, with all bands preferably being equal in size. Furthermore, the larger the image, the greater the startup time required before ~~beginning~~encoding can begin. This limitation is caused by the ~~conversion of the image data to encoded data, because the larger the image, the more~~increased pre-~~conversion~~encoding processing ~~required~~for larger images. Additionally, the more objects included in the image, the more memory that is required.

[0036] In ~~a~~the second mode of operation, ~~sometimes referred to as a~~ hereinafter "page assembly mode," the RIP 1050 receives, as multiple smaller images, an 80 inch ~~times.~~ x 50 inch color image having 2400 dots per inch. In this case, one of the multiple images~~, which~~ is primarily white. This image might be ~~characterized as~~ a template image~~, includes~~ and include information such as registration marks, color gradients, and identification marks~~, but is primarily white. Each of the~~ The other ~~of the multiple~~ images could, for example, be the images for pages of a magazine, each image being a separate page ~~of a magazine~~. Here, the RIP 1050 may be operated to ~~convert~~encode the image data representing the ~~entire~~ template image ~~into encoded data~~ as one job and to ~~convert~~encode ~~all of image~~the data ~~representing~~of the other ~~of the multiple~~ images ~~into image data in~~as another job. When ~~fully converted~~both jobs are complete, the ~~multiple images~~entirety of the image will be ~~represented by~~ encoded ~~data~~.

~~More particularly, in~~**[0037]** In the page assembly mode, the image is divided into page assemblies~~,~~. ~~o~~One of ~~which~~the pages is a ~~primarily white~~ template image ~~which~~that is, primarily white, and that is typically processed ~~by the RIP processor 1050a~~ without being split into bands. The other ~~of the multiple images are~~pages, however, are typically sliced into bands prior to being encoded by the RIP 1050. Because the area of each of the ~~other multiple~~page images is much smaller than the area of the entire image ~~discussed with reference to~~used in the first mode of operation, fewer bands are required ~~and, as~~. As a whole, it will take less time to ~~convert~~encode the image data representing the multiple images ~~into encoded data~~ in the page assembly mode than the time required to ~~convert~~encode the image data representing the entire image ~~into image data~~ in the band~~ing~~ed mode discussed above. Thus, the RIP processor 1050a ~~converts~~encodes the image data for each of the bands, ~~for~~in each of the ~~other multiple images into encoded data~~non-template pages, in a separate operational process. The job, or jobs if the template image is pre-processed, is completed only after ~~the multiple operational processes are performed to convert~~ all of the ~~image data representing the multiple images forming~~pages, which together represent the entire image, ~~into~~are encoded~~ data~~.

[0038] Although, the image discussed ~~with reference to~~in the page assembly mode description may be the same ~~as the~~ image discussed ~~with reference to~~in the ~~prior page assembly~~banded mode description, ~~conversion~~encoding in the page assembly mode will typically result in ~~even~~ a greater amount of encoded data than ~~the conversion~~encoding in the ~~previously discussed~~ banding mode. For example, the RIP 1050 may generate two gigabytes of encoded data ~~may be generated by the RIP 1050 to represent the image~~ in the band~~ing~~ed mode, ~~while~~yet generate three gigabytes of ~~image~~encoded data ~~could be generated by the RIP 1050 to represent~~for the same image in the page assembly mode ~~because there would be~~. The discrepancy is due to the page assembly mode retaining more uncompressed data. ~~Further, whether the banding or page assembly modes are utilized by the RIP 1050, the entire image cannot be converted into encoded data in a single operational process due to processing power limitations of the RIP 1050.~~

[0039] In the band~~ing~~ed mode, the image bands may be satisfactorily converted using ~~a~~the ~~LZW~~PB technique. In the page assembly mode, a template image~~, of say 16 megabytes,~~ may be satisfactorily converted using a PB technique since it is primarily white or primarily black and so is made up of mainly a repeating stream of 0s and 1s, respectively. However, the PB technique will often produce unsatisfactory results ~~if~~when used to convert the bands of the other of the multiple images. Accordingly, in the page assembly mode, these bands are converted using an LZW technique. Thus, in the page assembly mode, different compression techniques are utilized for a single image and perhaps even in a single job.

~~Accordingly, in the first~~**[0040]** Referring to FIG. 4, accordingly, in one embodiment of the ~~present~~ invention, the RIP 1050 ~~is selectively operable~~can operate in either the band~~inged~~ed or the page assembly mode ~~operation. Hence, in operation, the~~ The RIP 1050 initially scans the received image data ~~representing the image, or image bands if the bands are sliced during pre-processing,~~ to determine if band~~inged~~ed mode or page assembly mode operations ~~is~~are ~~required~~appropriate. Alternatively, the image may be sliced into bands during pre-processing. (STEP 3000). If ~~it is determined~~the RIP 1050 determines (STEP 3010) that band~~inged~~ed mode ~~operation is required, the RIP 1050 implements an LZW~~is appropriate, it encodes the image data using the PB technique ~~to convert the image data into encoded data.~~ (STEP 3020). If~~, on the other hand, it is determined~~, however, the RIP 1050 determines that page assembly mode ~~operation is required,~~is appropriate, it uses a different technique (STEP 3030). Referring to FIG. 5, the RIP 1050 further determines if the page image data represents a template image or banded image (STEP 3050). If ~~it is determined that~~ the page image data represents a template image, which as described is likely to be primarily black or white, the RIP 1050 ~~implements~~uses ~~a~~the PB technique to ~~convert~~encode the template image ~~into encoded data~~(STEP 3020). If, however, ~~it is determined that~~ the page image data represents a banded image, the RIP 1050 ~~implements~~uses ~~a~~the LZW technique to ~~convert~~encode the banded image data ~~into encoded data~~(STEP 3060). The selective operation of the RIP 1050, ~~depending on~~in response to the ~~received~~type of image data received, facilitates ~~the~~a more efficient and effective processing of ~~different types of~~ large images than ~~has been~~was previously obtain~~abled~~ed in conventional RIPs.

~~According to~~**[0041]** In a second embodiment of the ~~present~~ invention, ~~as~~encoding can be interrupted and a more efficient compression technique may be applied. The invention chooses to interrupt the processing if the stream contains a section of all black or all white data. As a stream of sequential data is processed prior to encoding, if~~, at the start of the sequence,~~ the immediately preceding character, which ~~is~~has yet to be encoded, matches the next character in the stream, and this next character is either solid black ~~or solid white~~(e.g., ~~and hence digitally represented in binary form by~~a stream of all 1's), or solid white (e.g., a stream of all 0's), encoding is interrupted. During the interruption, a determination is made as to whether the invention determines if one or more characters, immediately following the next character in the sequence, also match the next character.

~~The second~~[0042] Another embodiment of the invention will now be described with reference to FIG. 2. As shown, FIG. 2 represents a ~~somewhat~~ simplified, exemplary depiction of an image processing system 2000. The system 2000 ~~includes~~comprises a raster image processor~~(~~, hereinafter "RIP~~)~~," 2050, ~~which~~an imager controller 2100, and an imager 1150. The RIP 2050 receives an image and ~~converts~~encodes the image ~~into encoded~~ data. The encoded data is then transmitted to imager controller 2100, which generates imager control signals ~~to the imager 1150 in accordance with the encoded~~based on decoded data received from the RIP ~~2050 to control the~~2050. The imager 1150 ~~after decoding the received data~~in FIG. ~~This imager 1150~~2 is identical to the imager 1150 of FIG. 1. ~~More particularly~~Specifically, the control signals from the imager controller processor 2100a control the operation of the imager scanning assembly 1150a ~~so as~~ to form the image on a medium 1150c~~,~~. The ~~which~~medium could be identical to the medium 1150c in FIG. 1. The medium 1150c is supported within the imager 1150 of FIG. 2. As shown, the imager 1150 includes a cylindrical drum 1150b for supporting the medium 1150c.

[0043] In ~~the second~~this embodiment of the present invention, the RIP processor 2050a implements a compression technique, which will hereafter be referred to as the "*AGFA* technique." The *AGFA* ~~compression~~ technique~~. Using the **AGFA** compression technique, variable length of~~ can process strings of byte based data ~~can be processed. Processing using the **AGFA** technique will be~~of variable length. Using the *AGFA* technique is substantially faster then the LZW compression technique for many large image applications~~ than the LZW compression techniques~~, while still providing satisfactory results for both color images as well as those ~~which~~that are primarily black -and -white. Furthermore, the *AGFA* technique is not limited to single bytes of data, and is therefore capable of compressing data on an arbitrary pixel or bit boundary basis. Additionally, the *AGFA* technique is capable of providing a higher compression rate than ~~both~~either the PB ~~and~~or the LZW compression techniques implemented separately.

[0044] An exemplary representation of a stream of sequential input data as it would appear entering a RIP processor 2050a prior to encoding could, for example, include the string of characters "~~“~~abc0 . . . 01c01"~~”~~. The string "~~“~~0 . . . 0"~~”~~ is a large string of ~~zero's, for example representing image information for 32 k pixels~~zeros.

[0045] Using the *AGFA* technique, the encoding and decoding processors, i.e. the RIP processor 2050a and imager controller processor 2100a, must coordinate ~~on~~ the transmission and receipt of codes, similar to the coordination required by the LZW technique~~s~~. However, as will be described~~ further~~ below, the *AGFA* technique uses a compression dictionary containing four pre-defined compression codes. The characters in the input string are scanned to determine if a scanned sub-string of characters match certain ~~of these~~ pre-defined compression codes. If so, the matching sub-string of characters is encoded with the applicable pre-defined compression code. If a sub-string of characters does not match a pre-existing code, a new code~~s~~ corresponding to the sub-string~~s~~ ~~are~~is added to the dictionary.

~~Further~~**[0046]** Furthermore, the *AGFA* technique provides a look-ahead function~~, in which to determine whether or not~~. The look-ahead function determines if the sub-string is greater than a minimum number, preferably ~~6,~~6 bytes, and if ~~so~~ the sub-string is encoded with a new code~~, which includes~~; the new code comprising any applicable pre-existing code~~,~~ and the length of the code field. The length of the code field is the width of the pre-existing code, with th~~is~~e ~~code~~length forming the most significant bits and serving as a continuation indicator~~, and any~~. Any new ~~coding, with this coding~~code follows the length; the new code forming the least significant bits. Like the LZW technique~~s~~, sub-strings are initially defined by codes having 9 bits~~ or digits~~, but may be increased to up to 12 bits to add new codes. Once the 12 -bit limit is exceeded, the dictionary is reset and subsequent codes are again defined initially with 9 bits.

[0047] Referring to FIG. 3, in the *AGFA* technique, four codes are predefined and stored in the code dictionary ~~3000 on~~1300 in the RIP's memory 2050b as codes 1330. In the present example these codes are: the code 100, representing a sub-string of all ~~zero bytes~~zeroes, which corresponds to solid white~~,~~; the code 101, representing a sub-string of all ~~one bytes~~ones, which corresponds to solid black~~,~~; the code 102, representing a reset~~,~~; and the code 103, representing ~~an~~the end of the compressed encoded data. In the present example, codes 104, 105, ~~and 106~~106, etc. represent sub-strings of new patterns which are generated during the processing of the input dat~~e~~a and also stored ~~on~~in the RIP's memory 2050b in the code dictionary ~~3000.~~ 1300. It will be recognized that because codes for the strings corresponding to white and black are partially predefined~~, reduced processing is required to generate these codes, since the~~. Since predefined codes can simply be read by the RIP processor 2050a from the dictionary~~ codes as~~, reduced processing is required to generate these codes.

[0048] Using the *AGFA* technique, the RIP processor 2050a first sets a reset code 102 (read from the code dictionary ~~3000~~1300) and reads the ~~"~~“a~~"~~” in the sequence and the ~~"~~“b~~"~~” immediately thereafter~~ in the sequence~~. The RIP processor 2050a then determines from the code dictionary ~~3000,~~1300, if a code exists for the character sequence ~~"~~“ab~~"~~”. Since, in this example, no such code exists at this point in the processing, a new code 105 is generated to represent the new pattern string ~~"~~“ab~~"~~” and the new code is stored in the code dictionary ~~3000~~1300 ~~on~~in memory 2050b. The RIP processor 2050a continues ~~by~~and read~~ings~~s the ~~"~~“c~~"~~” immediately following the ~~"~~“b~~"~~” in the sequence. The RIP processor 2050~~b~~a determines if a code exists for the character sequence ~~"~~“bc~~"~~”. Since, in this example, no such code exist at this point in the processing, a new code 106 is generated to represent the new pattern string ~~"~~“bc~~"~~” and the new code is stored in the dictionary ~~3000.~~1300.

[0049] The RIP 2050 continues ~~by~~and read~~ings~~s the ~~"~~“0~~"~~” immediately following the ~~"~~“c~~"~~” in the sequence. The RIP processor 2050a determines if a code exists for the character sequence ~~"~~“c0~~"~~”. Since, in this example, no such code exists at this point in the processing, a new code 107 is generated to represent the new pattern string ~~"~~“c0~~"~~” and the new code is stored in the dictionary ~~3000.~~ 1300. Also, because the ~~"~~“0~~"~~” is recognized as a special character, the RIP processor 2050a~~,~~ automatically scans ahead to read the next character ~~in the sequence~~ to determine if it matches ~~with~~ the initial ~~"~~“0~~"~~”. ~~in~~If the ~~sequence. If~~next character is not~~,~~ a matching “0,” the scanning ahead is immediately discontinued and the RIP processor 2050a proceeds with normal processing. If ~~so,~~the next character is a matching “0,” the scanning ahead continues on~~ a~~, character by character~~ basis~~, until no ~~match with "~~matching “0"” is found~~, at which~~. At that point, the scanning ahead is discontinued and normal processing continues.

[0050] In this exemplary application of the *AGFA* technique, the RIP processor 2050a scans ahead and counts the number of "repeated “0"” or "“1"” bytes in the sequence. Preferably, a compression threshold is pre-established and stored ~~on~~in the RIP memory 2050b. For example, the threshold might correspond to a 4 to 1 compression rate. If such a threshold is utilized, and the number of "repeated “0"” or "“1"” bytes counted is less than the number required to meet or exceed the threshold, e.g. if the sequence consists of only one or two zeros or ones, then a new code would be established for the sequence in the normal manner. Only if the number of "repeated “0"” or "“1"” bytes counted meets or exceeds the threshold~~,~~ is the sequence encoded using the applicable pre-defined code 100 or 101.

[0051] Assuming in the present example that the number of "“0"” bytes counted by the RIP processor 2050a meets or exceeds the threshold, the bits in the “count ~~is determined to be~~position” represent a repeat count. Either 9, 10, 11, or 12 bits can be used to code the repeat count. However, if the count is so great that more than 11 bits would be required for the encoding, a continue code which may be generated by the processor 2050a or retrieved from memory 2050b, is inserted as the least significant bit in the output code~~ to enable~~. This continue code enables the output codes representing the entire sequence of zeros or ones to be strung together. ~~Accordingly~~Therefore, no matter how long the sequence is, the low or leasst significant bit of each output code within the string of output codes would represent either an end code, or a continuation of the coding. Hence, 1 bit is sacrificed for the end/continuation bit leaving 8, 9, 10, or 11 bits for the repeat count.

[0052] Accordingly, in the present example, the output code for the repeat count of "“0"” characters would be formed ~~with~~using the code "“100"” to indicate that this is a sequence of "“0"” characters, followed by "“102"” representing a first portion of the repeat count, and "“001"” indicating that the output codes for the repeat count continues. Thus, the first code in the string of repeat count output codes would be "“100102001"”. The second code in the string of repeat count output codes could be ~~"102001",~~“102001,” with the "“102"” representing a second portion of the repeat count, and "“001"” indicating that the output codes for the repeat count continues. The last code in the string of repeat count output codes could be "“0201"”. The high bit of the last output code "“0201~~" is made clear to indicate~~” indicates that this is the end of the repeat count information in this field.

[0053] Using the repeat count multiple ~~output codes~~times, the strung -together codes for the entire repeat count would, in the above example be "“1001020011020010201"”. Thus, the strung together multiple bytes of output codes provide a full representation of the repeat count. In practice, five output codes may be used to represent up to four billion characters. Notwithstanding the number of bits in the output codes, the high bit is used to represent the count. Accordingly, whatever output code size is used, full advantage is taken of all available bits for the repeat count.

~~It is perhaps worthwhile emphasizing here that conventional~~[0054] Conventional LZW techniques lack the ability to scan ahead. Conventional PB techniques, on the other hand, scan ahead to locate matches with whatever character has been read and must fully generate the match coding for each matching sequence. In contrast to both, the present invention scans ahead to locate matches with only selective characters, preferably only white and black, respectively represented herein by "“0"” and "“1"”. Furthermore, the ~~present~~ invention ~~uses~~can use a predefined code for each of the selected characters, e.g. white and black. Hence, ~~and hence the match~~ coding for each matching sequence need only be partially generated~~,~~ since ~~the~~ predefined codes, e.g. codes 100 or 101, ~~which identifies the applicable sequence as a sequence of white or black characters is~~are pre-generated and need only be read from the code dictionary ~~3000.~~ 1300. Accordingly, the present invention is capable of providing superior encoding of~~, for example,~~ large images using less computing resources and computing time.

[0055] As noted above, once the RIP processor 2050a determines it is at the last "“0"” in the sequence, i.e. by determining from ~~the~~ scanning ahead ~~on a character by character basis~~ that ~~a~~the next character does not match ~~with~~a ~~"0",~~“0,” the scanning ahead is discontinued and normal processing ~~continues~~resumes. Thus, the RIP processor 2050a continues by reading the "“1"” immediately following the last "“0"” in the sequence. The processor 2050a determines if a code exists for the character sequence "“01"”. Since, in this example, no such code exist at this point in the processing, a new code 108 is generated to represent the new pattern string "“01"~~.~~” and the new code is added to the dictionary 1300. The processor 2050a proceeds by reading the "“c"” immediately following the "“1"” in the sequence. The processor determines if a code exists for the character sequence "“1c"”. Since, in this example, no such code exists at this point in the processing, a new code 109 is generated to represent the new pattern string "“1c"~~.~~” and the new code is added to the dictionary 1300.

[0056] The processor ~~further~~2050a then proceeds by reading the "“0"” immediately following the second "“c"” in the sequence. The processor 2050a determines if a code exists for the character sequence "“c0"”. In this example, such a code, i.e. code 107, does exist. The RIP processor 2050a ~~also~~then scans ahead to determine if another "“0"” immediately follows this occurrence of "“c0"”. Since, in this case the ~~RIP processor 2050a determination~~next character is ~~negative,~~not a “0,” the scanning ahead is discontinued and normal processing ~~continues~~resumes.

[0057] The processor 2050a ~~now~~ proceeds by reading the "“1"” immediately following the second "“c0"” in the sequence. The processor 2050a determines if a code exists for the character sequence "“c01"”. Since, in this example, such a code does not exist, a new code 110 is generated to represent the new pattern string "“c01" ~~which~~” and the new code is added to the code dictionary 1300. Because the “1” is the last character, the combination of the last generated code and the last character can be represented as "“1071"”. The RIP processor 2050a also scans ahead to determine if another "“1"” immediately follows this occurrence of "“c01"”. Since, in this case the ~~RIP processor 2050a determination~~next character is ~~negative,~~not a “1,” the scanning ahead is discontinued~~ and normal~~. Normal processing would ~~continue~~resume if further characters remained to be encoded. However, since ~~the~~ "“c01"” and 1 are the final characters, encoding ends.

[0058] The processor 2050a ultimately generates encoded output data ~~forming a~~of the form "102abc10010200110200102011071103". The sequence ~~including the~~ includes the encoded string ~~of characters "102abc1001020011020010201107- 1103".~~ and an end code, code 103.

[0059] Similar to the LZW techniques, in the *AGFA* technique~~,~~ the RIP processor 2050a builds a tree of numerous codes ~~generated~~ using a combination of pre-defined ~~or other codes~~ and generated codes. The *AGFA* technique is thereby ~~is~~ capable of providing satisfactory results even though the processing is performed on a byte by byte basis to find repeating bytes. ~~However, as compared~~ Compared to ~~LZW techniques, in~~ the ~~***AGFA***~~LZW technique, the *AGFA* technique requires substantially reduced processing time and resources ~~required by the RIP 2050~~ to encode large sequences ~~is substantially reduced through the use of~~because it uses special pre-defined codes. The decoding processor, i.e. the imager controller processor 2100a~~, which could serve as a printer controller (not shown) or be some other type decoding device,~~ builds a similar tree using the codes in the code dictionary received from the encoding processor, i.e. the RIP processor 2050a. ~~Basically~~ Aside from the imager controller processor 2100a, the decoding processor could serve as a printer controller (not shown) or be some other type of decoding device. The decoding processor performs the reciprocal of the encoding process to decode the encoded sequence characters ~~"~~"102abc10010200110200102011071103~~"~~". It should be understood that the encoded data could if desired be transmitted to the decoding device via a direct communications link, a local network, a public network such as the Internet, or some other type of network. Further, such communications may be by wire communications or wireless communications. It will also be recognized by those skilled in the art that, while the invention has been described above in terms of one or more preferred embodiments, it is not limited thereto. Various features and aspects of the above described invention may be used individually or jointly. Furthermore, although the invention has been described in the context of its implementation in a particular environment and for particular purposes, e.g. imaging, those skilled in the art will recognize that its usefulness is not limited thereto and that the present invention ~~can~~may be beneficially utilized in any number of environments and implementations. Accordingly, the claims set forth below should be construed in view of the full breadth and spirit of the invention as disclosed herein.